



INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA (ISEL)

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA (DEI)

LEIM

LICENCIATURA EM ENGENHARIA INFORMÁTICA E MULTIMÉDIA
UNIDADE CURRICULAR DE PROJETO

AI Games Platform



Ana Carriço (50725)

Gabriel Rodrigues (50787)

Orientador(es)

Professor Doutor Artur Ferreira

Professor Doutor Jorge Nuno Silva (FCUL)

Julho, 2025

Resumo

Nos dias de hoje, assistimos à crescente digitalização de jogos clássicos como xadrez, damas e outros jogos de tabuleiro tradicionais que têm encontrado uma nova vida no mundo virtual. No entanto, muitas plataformas existentes ou não integram sistemas de inteligência artificial ajustados ao nível dos jogadores, ou oferecem uma seleção limitada de jogos disponíveis, comprometendo a experiência do utilizador.

Este projeto trata da implementação de uma plataforma de jogos onde os utilizadores possam competir tanto contra outros jogadores humanos como contra sistemas de inteligência artificial. Desenvolvida com Unity, php, MySQL e tecnologias web como HTML, CSS e JavaScript a plataforma visa responder a diferentes objetivos, como entretenimento, aprendizagem ou competição. Além do jogo em si, oferece funcionalidades adicionais como a revisão de partidas anteriores, a visualização de um perfil personalizado com estatísticas e a atribuição de conquistas, promovendo o interesse pela plataforma e a competitividade.

A avaliação experimental realizada demonstraram a estabilidade da plataforma e a eficácia da integração da inteligência artificial com diferentes níveis de dificuldade. A interface revelou-se acessível e intuitiva, respondendo positivamente às expectativas de utilizadores com diferentes perfis.

Palavras Chave: inteligência artificial; jogo do galo; Monte Carlo Tree Search; plataforma de jogos; quatro em linha.

Abstract

Nowadays, we are witnessing the growing digitization of classic games such as chess, checkers, and other traditional board games, which have found new life in the virtual world. However, many existing platforms either fail to integrate artificial intelligence systems that adapt to the players' skill levels or offer a limited selection of available games, compromising the user experience.

This project concerns the implementation of a gaming platform where users can compete either against other human players or against artificial intelligence systems. Developed using Unity, PHP, MySQL, and web technologies such as HTML, CSS, and JavaScript, the platform aims to address various goals, including entertainment, learning, and competition. In addition to the gameplay itself, it offers extra features such as reviewing past matches, viewing a personalized profile with statistics, and earning achievements, which help promote engagement with the platform and encourage a competitive spirit.

The experimental evaluation conducted demonstrated the platform's stability and the effectiveness of integrating artificial intelligence with varying difficulty levels. The interface proved to be accessible and intuitive, meeting the expectations of users with different profiles.

Keywords: artificial intelligence; connect four; gaming platform; Monte Carlo Tree Search; tic tac toe.

Agradecimentos

Gostaríamos de agradecer a todos os que ajudaram no desenvolvimento do projeto, com ênfase especial ao orientador do projeto Artur Ferreira. Agradecemos também a todas as pessoas que participaram nos testes de experiência de utilizador e sugeriram ideias de melhoria.

Índice

Resumo	i
Abstract	iii
Agradecimentos	v
Lista de Tabelas	ix
Lista de Figuras	xi
1 Introdução	1
2 Trabalho Relacionado	3
3 Modelo Proposto	7
3.1 Requisitos	7
3.2 Fundamentos	9
3.2.1 <i>Frontend</i>	9
3.2.2 Lógica dos Jogos	10
3.2.3 Conexão entre Jogadores	10
3.2.4 Base de Dados	11
3.2.5 Servidor	11
3.3 Abordagem	11
3.3.1 <i>Frontend</i>	11
3.3.2 Algoritmo de Inteligência Artificial	12
3.3.3 Base de Dados	13
3.3.4 Servidor	14

4 Implementação do Modelo	17
4.1 Implementação do <i>Frontend</i>	18
4.2 Implementação dos Jogos e da IA	18
4.2.1 Interface e Lógica	18
4.2.2 Revisão de Jogos	19
4.2.3 Conexão Entre Dois Jogadores	20
4.2.4 Inteligência Artificial	21
4.2.5 Fluxo do Jogo	23
4.3 Implementação do <i>Backend</i>	23
4.3.1 Estrutura Geral	23
4.3.2 Processamento de Dados	23
4.3.3 Integração com Unity e HTML	24
4.3.4 Estrutura da Base de Dados	25
5 Validação e Testes	27
5.1 Funcionalidades Essenciais	27
5.2 Jogo do Galo	29
5.3 Quatro em Linha	30
5.4 Inquérito aos Jogadores	30
6 Conclusões e Trabalho Futuro	33
A Regras dos Jogos	35
A.1 Jogo do Galo	35
A.2 Quatro em Linha	36
Bibliografia	37

Lista de Tabelas

3.1	Requisitos funcionais e não funcionais	8
4.1	Tempos Típicos para os Níveis de Dificuldade em Cada Jogo .	22
5.1	Resultados dos Testes do Jogo do Galo	30
5.2	Resultados dos Testes do Quatro em Linha	30

Lista de Figuras

2.1	Chess.com	4
2.2	Revisão de partidas do Chess.com	4
2.3	Tabletopia	4
2.4	Board Game Arena	5
3.1	Casos de Utilização do Sistema	8
3.2	Diagrama de Arquitetura	9
3.3	Diagrama de Arquitetura das partidas	10
3.4	Diagrama de Fluxo do Servidor	11
3.5	Unity Cloud Dashboard	12
3.6	Modelo Entidade-Associação	15
4.1	Página Inicial	18
4.2	JSON de Movimentos de um Jogo	19
4.3	Quatro em Linha em Modo de Dois Jogadores	21
4.4	Diagrama de atividades da partida	23
4.5	Interface do Jogo do Galo no Website	24
4.6	Interface do Quatro em Linha no Website	25
4.7	Base de Dados das Partidas	25
5.1	Escolha da Dificuldade da IA	28
5.2	Atribuição de Conquistas	29
5.3	Quadro de Honra do Jogo do Galo	29
5.4	Preferência dos utilizadores pelos jogos disponíveis na plataforma	31
5.5	Percentagem de utilizadores que acharam a experiência de jogo clara e intuitiva	31
5.6	Avaliação da funcionalidade de revisão de jogos.	32

5.7 Correspondência entre dificuldade escolhida e expectativas dos jogadores	32
----------------------------------------------------------------------------------------	----

Capítulo 1

Introdução

Os jogos, ao longo das décadas, têm sido uma ferramenta valiosa não só para entretenimento, mas também para o desenvolvimento de competências cognitivas e investigação científica. Pela sua estrutura com regras claras e objetivos definidos, são ideais para promover o raciocínio lógico e a aprendizagem. Além disso, têm aplicação em áreas como a educação, psicologia, economia e ciência da computação. No entanto, o uso excessivo pode ter efeitos negativos, como a dependência, o isolamento social ou a redução do desempenho académico e profissional, reforçando a importância de um uso equilibrado.

O uso de técnicas de inteligência artificial no contexto dos jogos tem despertado crescente interesse e obtido resultados significativos nas últimas décadas. Um marco histórico nesse percurso foi a vitória do sistema *Deep Blue* da IBM sobre o campeão mundial de xadrez Garry Kasparov em 1997, demonstrando o potencial da IA em ambientes altamente estratégicos. Livros de referência como [Russell e Norvig, 2010] apresentam os jogos como domínio fundamental para a investigação e ensino da IA, descrevendo-os como um espaço ideal para aplicar algoritmos como Minimax, Monte Carlo Tree Search ou técnicas de aprendizagem por reforço.

Este projeto incide sobre o desenvolvimento de uma plataforma de jogos de informação perfeita, nos quais todos os jogadores têm total conhecimento do estado do jogo e não existem fatores de aleatoriedade, assim como descrito no livro [Neto e Silva, 2004]. Jogos como o poker, a sueca e a batalha naval não foram opções de implementação por não apresentarem, a todos os jogadores, conhecimento total do estado do jogo. Com isto em mente,

optou-se inicialmente pelo jogo do galo devido à sua simplicidade e diminuto espaço de soluções. Posteriormente, implementou-se o 4 em linha por ser mais complexo e permitir testar as capacidades do algoritmo num ambiente com regras diferentes (aplicação da gravidade), pondo à prova a escalabilidade do sistema.

Além da vertente competitiva, foi também implementado um sistema de revisão de jogos, com carregamento e reprodução de jogadas previamente registadas. Esta funcionalidade é particularmente útil para análise pós-jogo, sendo relevante tanto em contextos educativos como em ambientes de treino e avaliação de estratégias.

Este relatório está estruturado em capítulos que abordam os diversos aspectos do desenvolvimento da plataforma. O Capítulo 2 detalha o trabalho relacionado. Os fundamentos teóricos, requisitos e abordagens que orientaram o projeto são tratados no Capítulo 3, as decisões de implementação no Capítulo 4 e os testes realizados para validação funcional no Capítulo 5. Por fim, são discutidos os resultados obtidos e propostas de melhorias futuras que podem elevar a qualidade da aplicação no Capítulo 6. Posteriormente, as regras dos jogos utilizados são referidas no Apêndice A.

Capítulo 2

Trabalho Relacionado

Analizando as soluções existentes, identifica-se a escassez de plataformas de jogos de tabuleiro em digital que promovam a competitividade entre jogadores e apresentem um público-alvo mais abrangente ao constituir uma interface simples.

Plataformas como Chess.com [Aficionado, 2025] (figura 2.1) destacam-se pela usabilidade e pela qualidade da experiência oferecida no domínio específico do xadrez, proporcionando funcionalidades de treino, competição e análise estatística (figura 2.2). No entanto, estas soluções apresentam uma forte especialização, não suportando uma diversidade significativa de jogos. Por outro lado, plataformas como Tabletopia [Bokarev, 2025] e Board Game Arena [Isabelli e Colin, 2025] (figuras 2.3 e 2.4 respetivamente) oferecem um catálogo abrangente de jogos, mas fazem pouco ou nenhum uso de inteligência artificial nos seus sistemas. Esta limitação representa uma barreira à entrada de utilizadores menos experientes, que muitas vezes preferem treinar ou explorar o jogo a um ritmo próprio antes de enfrentar adversários humanos.

Para além dos clássicos como o xadrez ou as damas, existem jogos de informação perfeita menos conhecidos, mas igualmente relevantes do ponto de vista estratégico, como o Amazonas, o SanQi, ou o Hex. Estes jogos mantêm as características de ausência de aleatoriedade e de informação oculta, mas são menos populares junto do público geral, sendo mais frequentes em contextos académicos ou de investigação em Inteligência Artificial.



Figura 2.1: Chess.com



Figura 2.2: Revisão de partidas do Chess.com

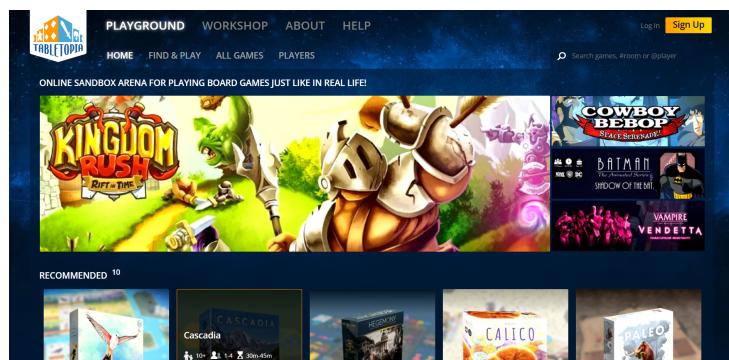


Figura 2.3: Tabletopia



Figura 2.4: Board Game Arena

Capítulo 3

Modelo Proposto

Este capítulo apresenta a estrutura do modelo conceitual desenvolvido para a execução do projeto.

Serão explorados os requisitos pretendidos (secção 3.1), tanto funcionais como não funcionais. De acordo com esses requisitos, foram definidos casos de utilização que demonstram as diferentes funções que o utilizador pode desempenhar no sistema.

Em seguida, são apresentados os fundamentos teóricos e tecnológicos (secção 3.2) de forma a sustentar as tecnologias do sistema e contextualizar a comunicação entre os vários componentes.

Por fim, a secção da abordagem (secção 3.3) apresenta as tecnologias escolhidas e porque foram utilizadas, desde linguagens de programação a bibliotecas.

3.1 Requisitos

Esta aplicação tem como objetivo disponibilizar uma plataforma online na qual os utilizadores possam aceder aos jogos desenvolvidos, com suporte a *backend* e armazenamento numa base de dados. De seguida, apresentam-se os requisitos funcionais, correspondentes às funcionalidades diretamente percecionadas pelo utilizador, e requisitos não funcionais que não descrevem funcionalidades diretas, mas sim qualidades, restrições ou características técnicas.

A tabela 3.1 apresenta os diferentes requisitos do projeto.

Tabela 3.1: Requisitos funcionais e não funcionais

Tipo de Requisito	Descrição
Funcional	Autenticar o utilizador Implementar jogo do galo Implementar jogo quatro em linha Permitir jogar contra IA de diferentes dificuldades Suportar jogos de dois jogadores Armazenar dados sobre jogos Apresentar quadro de honra Atribuir conquistas a jogadores Rever jogos anteriores Permitir jogar contra humano
Não Funcional	Assegurar a integridade dos dados Apresentar <i>User Interface</i> (UI) simples e apelativa Armazenar de forma segura os dados do utilizador

A partir destes requisitos definiram-se os casos de utilização possíveis, ilustrados na figura 3.1.

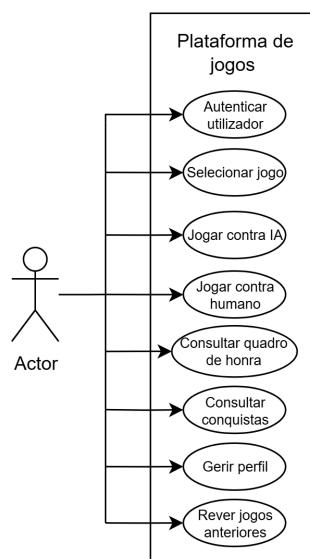


Figura 3.1: Casos de Utilização do Sistema

3.2 Fundamentos

O modelo proposto assenta na arquitetura cliente-servidor, sendo composto por uma interface web acessível ao utilizador (*frontend*), um sistema de processamento de pedidos (*backend*) e um sistema de armazenamento de dados (base de dados). A arquitetura cliente-servidor revela-se especialmente adequada ao problema, uma vez que permite separar claramente a lógica de apresentação da lógica de negócio e da gestão de dados. Esta separação facilita o desenvolvimento e a manutenção da aplicação, e o suporte a múltiplos utilizadores em simultâneo. Esta arquitetura é apresentada na figura 3.2. A seguir, referem-se as linguagens de programação e ferramentas aplicadas a cada componente.

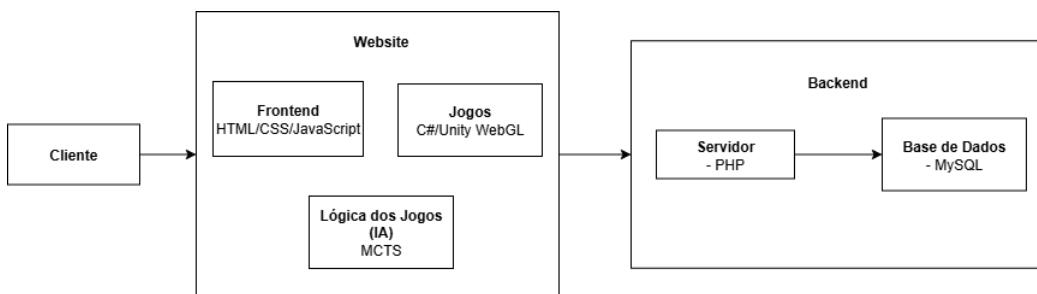


Figura 3.2: Diagrama de Arquitetura

3.2.1 *Frontend*

A interface gráfica do sistema foi desenvolvida utilizando Hypertext Markup Language (HTML), Cascade Style Sheets (CSS) e JavaScript, linguagens utilizadas no desenvolvimento de aplicações web responsivas.

A integração dos jogos no sistema segue um modelo no qual cada jogo funciona como uma componente independente da interface principal. Embora tenha sido considerada a possibilidade de desenvolver esta interface gráfica com HTML, CSS e JavaScript, a escolha final recaiu sobre o Unity, devido à sua capacidade de criar ambientes interativos e com boa performance gráfica. Para tornar os jogos acessíveis num navegador, a aplicação foi exportada em WebGL, uma tecnologia que permite compilar projetos Unity para serem executados diretamente numa página web, como explorado no tutorial disponível em [Technologies, 2017]. Esta abordagem facilitou a integração dos

jogos na plataforma online, permitindo que os utilizadores joguem diretamente no site.

3.2.2 Lógica dos Jogos

A lógica dos jogos, incluindo a Inteligência Artificial, foi implementada em C#. A IA foi desenvolvida a partir do algoritmo Monte Carlo Tree Search (MCTS) [AI e Games, 2018], escolhido pela sua capacidade de tomar decisões eficazes mesmo sem avaliação heurística explícita, adaptando-se bem a diferentes jogos.

3.2.3 Conexão entre Jogadores

Embora a arquitetura geral do sistema siga o modelo cliente-servidor, a componente de jogo multijogador adota uma abordagem *peer-to-peer*. Para isso, foram utilizados o Unity Relay e o Unity Lobby, que permitem que os jogadores se conectem diretamente entre si, mesmo estando em locais distintos. A figura 3.3 apresenta um diagrama de arquitetura de gestão das partidas.

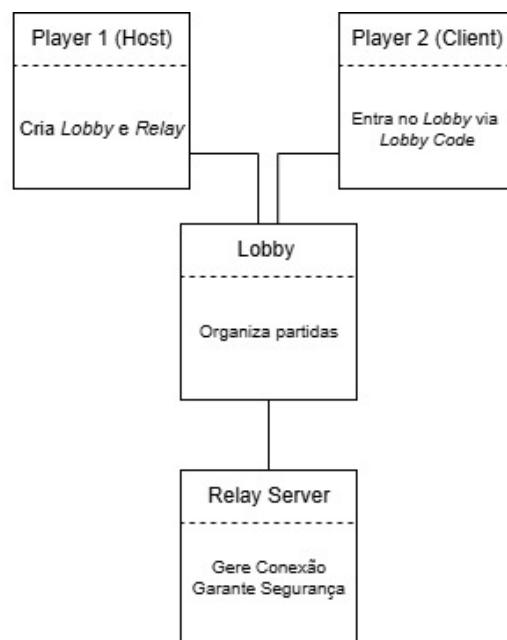


Figura 3.3: Diagrama de Arquitetura das partidas

3.2.4 Base de Dados

Para o armazenamento de dados dos jogos, utilizou-se uma base de dados suportada em MySQL dado o seu suporte a consultas complexas. A comunicação entre o servidor PHP e a base de dados foi feita através de interrogações Structured Query Language (SQL), permitindo um acesso eficiente e seguro.

3.2.5 Servidor

Utilizou-se o Cross-platform, Apache, MySQL/MariaDB, PHP e Perl (XAMPP), apresentado em [Ravikiran, 2025], para hospedar localmente o servidor PHP que funciona como *middleware*, facilitando a comunicação entre a interface gráfica desenvolvida no Unity e a base de dados MySQL. A troca de dados entre o Unity e o Hypertext Processor (PHP) é realizada através de requisições HTTP, com dados estruturados em formato JavaScript Object Notation (JSON). O fluxo desta abordagem, apresentado na figura 3.4, garante compatibilidade e facilita o processamento entre diferentes camadas da aplicação.

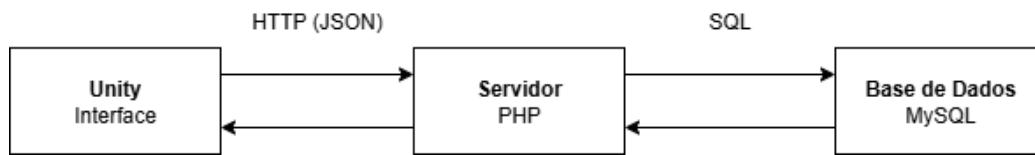


Figura 3.4: Diagrama de Fluxo do Servidor

3.3 Abordagem

A partir da arquitetura definida nos fundamentos e dos casos de utilização previamente estruturados, foi necessário escolher as linguagens de programação adequadas a cada componente do sistema.

3.3.1 *Frontend*

Para a interface gráfica do jogo foram utilizadas as ferramentas de edição de cenas do Unity, que permitem editar o aspetto visual dos jogos sem recorrer a folhas de estilo. O formato de exportação WebGL facilita a implementação da lógica e das interfaces gráficas dos jogos diretamente no navegador embutidas no HTML. Apesar de alternativas como HTML/CSS/JavaScript terem sido

consideradas para a interface, estas foram reservadas para a elaboração de elementos complementares da plataforma (como o quadro de honra ou *login*).

Além disso, o Unity oferece uma grande vantagem ao abstrair muitas das complexidades associadas à criação de jogos, incluindo a gestão da conexão entre jogadores com Unity Relay e Unity Lobby. Estes serviços facilitam a ligação entre jogadores num sistema *peer-to-peer* em tempo real e o seu uso/consumo pode ser consultado no Unity Cloud representado na figura 3.5.

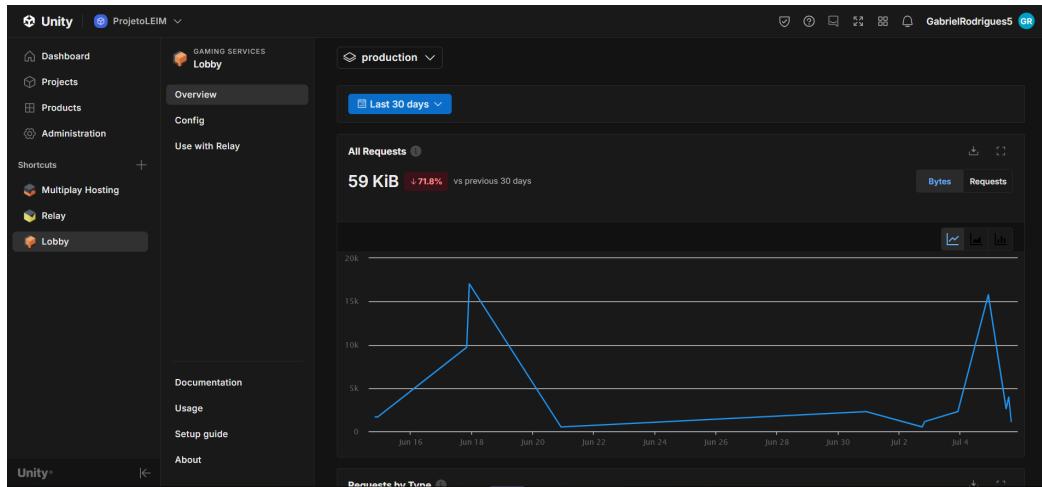


Figura 3.5: Unity Cloud Dashboard

Outra opção ponderada foi o uso da linguagem de programação Java para a lógica de jogos, Servlets, JavaServer Pages (JSP), a qual foi abandonada devido à complexidade da implementação da comunicação entre dispositivos em redes privadas ou atrás de Network Address Translation (NAT)/firewalls.

3.3.2 Algoritmo de Inteligência Artificial

O algoritmo de Inteligência Artificial (IA) que replica o comportamento de um jogador humano na plataforma foi desenvolvido no Unity. Inicialmente, foram considerados três algoritmos de IA:

- O algoritmo MiniMax com Alpha-Beta Pruning, apesar de simples e fácil de implementar, pode revelar-se computacionalmente custoso em jogos com muitos estados possíveis, apesar da otimização proporcionada pela poda. Tal como referido por [Russell e Norvig, 2010], "o

algoritmo Alpha-Beta continua a ter de explorar até atingir estados terminais, pelo menos, numa porção do espaço de estados. Esta profundidade normalmente não é prática, uma vez que se espera uma resposta num curto espaço de tempo (...)". Este atraso natural do processo torna a utilização menos fluida e piora a experiência do utilizador.

- O uso de um algoritmo de Deep Reinforcement Learning, como o oferecido pela biblioteca ML-Agents, tem a possibilidade de reduzir o custo computacional por jogada devido ao seu fator de aprendizagem. Contudo, exige grandes quantidades de dados para aprender estratégias eficazes e requer um modelo de recompensa bem estruturado.
- O algoritmo Monte Carlo Tree Search (MCTS) utiliza simulações aleatórias para tomar ações, acrescentando um elemento de falso realismo, uma vez que duas partidas consecutivas não decorrem necessariamente da mesma forma. Este algoritmo, mesmo não garantido jogadas ótimas, consegue, com tempo de pesquisa suficiente, fazer boas decisões que levam à vitória mesmo contra jogadores experientes. Além disso, o MCTS apresenta uma estrutura abstrata que pode ser adaptada a qualquer jogo e permite, com base no tempo de pesquisa, limitar a procura de resultados ótimos, criando assim vários níveis de dificuldade.

3.3.3 Base de Dados

Quanto à realização da base de dados, o MySQL foi a solução escolhida. Embora se tenha considerado o uso de eXtensible Markup Language (XML) como alternativa para armazenamento dos dados, optou-se por um Sistema de Gestão de Base de Dados (SGBD) relacional por oferecer vantagens significativas. O MySQL destaca-se pela sua eficiência no tratamento de grandes volumes de dados, integridade referencial, e mecanismos de consulta otimizados através de SQL. Em contraste, o XML, sendo um formato de marcação orientado a documentos, não oferece funcionalidades avançadas de gestão de dados, tornando-se mais complexo de manipular e difícil de escalar com o crescimento da aplicação. Assim, o MySQL revelou-se uma solução mais adequada aos objetivos do projeto.

A modelação de dados representada no Modelo Entidade–Associação (MEA) da figura 3.6 reflete diretamente os requisitos funcionais definidos.

A estrutura do modelo foi concebida de forma a dar resposta direta aos requisitos definidos para a plataforma. A entidade Game, que representa os diferentes jogos disponíveis, está ligada à entidade Match, assegurando que cada partida registada corresponde a um jogo específico, e à entidade Player, permitindo manter um sistema de *rating* independente por jogo, avaliando o desempenho dos jogadores. Por sua vez, a entidade Move foi incluída para garantir a possibilidade de reconstruir jogos, armazenando cada jogada individual com a informação necessária, o que é essencial para a funcionalidade de revisão de partidas.

3.3.4 Servidor

No que diz respeito à camada de servidor, utilizou-se PHP devido à sua compatibilidade com bases de dados MySQL e à familiaridade da equipa com o seu ecossistema.

A integração com o Unity foi feita por meio de UnityWebRequest, que envia e recebe dados em JSON, facilitando a serialização de dados. A estrutura de comunicação adotada permite, por exemplo, o envio de movimentos, criação de partidas e, registo de resultados.

Relativamente ao servidor, o uso de PHP foi considerado como a melhor forma de interligar a interface gráfica feita em Unity com o envio de informação para a base de dados, utilizando JSON.

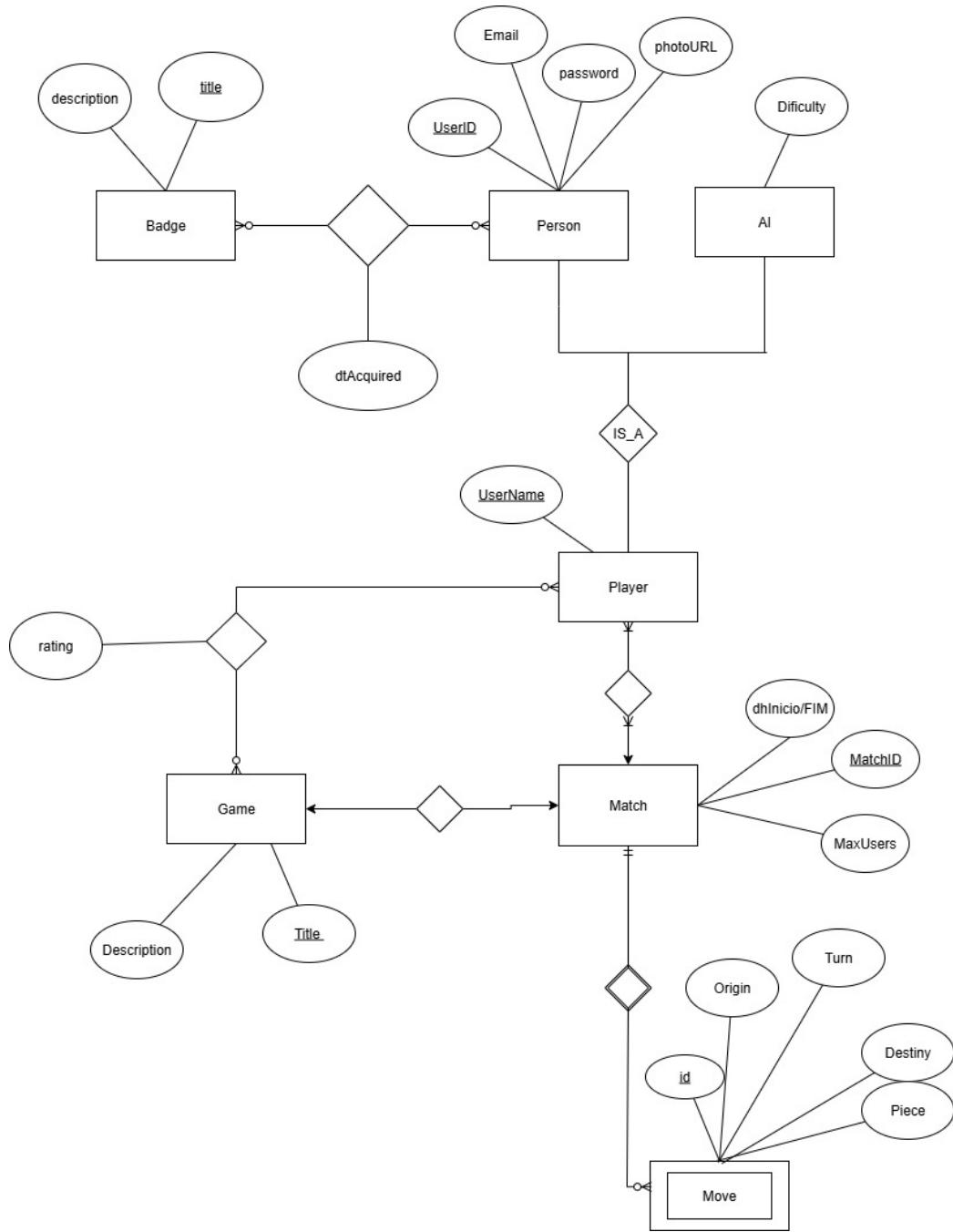


Figura 3.6: Modelo Entidade-Associação

Capítulo 4

Implementação do Modelo

Este capítulo descreve os detalhes da implementação prática do sistema desenvolvido, abordando as três principais camadas da aplicação: *frontend*, lógica dos jogos e da inteligência artificial, e *backend*.

Na secção 4.1, é apresentada a construção da interface gráfica baseada em tecnologias web (HTML, CSS e JavaScript), que permite uma experiência interativa e responsiva. É também explicado o processo de comunicação assíncrona com o servidor por meio de chamadas AJAX.

A Secção 4.3 detalha a implementação dos jogos em Unity, incluindo a lógica do jogo, a integração da inteligência artificial com o algoritmo Monte Carlo Tree Search e o sistema de revisão de partidas. A arquitetura baseada em herança facilita a expansão da plataforma para novos jogos. É ainda explorado o mecanismo de *multiplayer online* com o uso dos serviços de rede do Unity, como Authentication, Lobby, Relay e Netcode for GameObjects.

Por fim, na Secção 4.3 (relativa ao *backend*), discute-se a implementação do servidor utilizando PHP, responsável por receber e processar as requisições do *frontend* e do Unity. É também descrita a estrutura da base de dados em MySQL, bem como as medidas de segurança implementadas, como *hashing* de palavras-passe e uso de *prepared statements*.

O *website* do projeto encontra-se disponível em [Carriço e Rodrigues, 2025]. Pode ser necessário fornecer um certificado pois o uso de certificado SSL automático exige domínio privado e pago.

4.1 Implementação do *Frontend*

No *frontend*, a comunicação com o *backend* é realizada por meio de chamadas Asynchronous JavaScript and XML (AJAX), utilizando a Application Programming Interface (API) `fetch`. Estas chamadas permitem que a interface interaja dinamicamente com o servidor sem recarregar a página.

Visualmente, o HTML estrutura os elementos da interface, como visualização do perfil e jogos. O CSS é responsável pela aparência e estilos, como cores, fontes e posicionamento, mantendo uma interface coesa ao utilizar tons azulados, como apresentado na figura 4.1. O JavaScript gera a lógica de interação no lado do cliente, incluindo gestão de eventos, a preparação e envio de requisições via AJAX, e o processamento das respostas recebidas para atualizar dinamicamente a interface do utilizador.

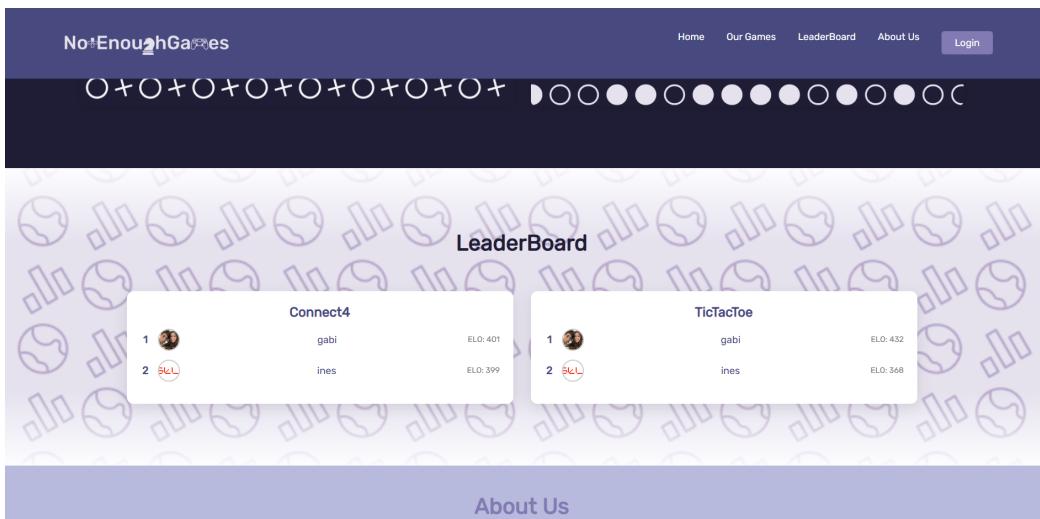


Figura 4.1: Página Inicial

4.2 Implementação dos Jogos e da IA

4.2.1 Interface e Lógica

Para a implementação inicial da interface gráfica e da lógica do jogo, utilizou-se como referência um tutorial [Oamen, 2023] sobre criação de um jogo baseado em turnos. O tutorial ajudou a estabelecer conceitos como o uso de

um *array* de *strings* para representar o estado do tabuleiro e a utilização de uma variável para controlar de quem é o turno.

A arquitetura do sistema foi desenhada com base numa classe abstrata que centraliza toda a lógica comum entre os diferentes jogos desenvolvidos. Esta classe gera o estado da rede, da comunicação com a base de dados e da interface, com a qual o utilizador comunica através de botões. Cada jogo estende esta classe e implementa os métodos abstratos responsáveis pela lógica específica do jogo, como verificação do vencedor e jogadas, facilitando a manutenção e a expansão do sistema.

Sempre que o utilizador não tem permissão para realizar uma ação, por exemplo, jogar na vez do oponente, esses botões ficam inativos, impedindo que interrompa o fluxo da partida.

4.2.2 Revisão de Jogos

O sistema de revisão de jogos permite ao utilizador analisar todas as jogadas realizadas numa partida anterior. Esse processo tem início com a leitura do *matchId* através de um pedido HTTP. O servidor responde com uma sequência de jogadas no formato JSON (ver figura 4.2), que é desserializado e cada jogada é aplicada sequencialmente ao tabuleiro anterior.

Após a execução de cada movimento, o estado atual do tabuleiro é armazenado numa lista interna que guarda todo o histórico da partida. Esse mecanismo possibilita navegar entre os diferentes estados do jogo, tanto para frente quanto para trás, até ao estado final.



```

Pretty-print □
{
  "moves": [
    {
      "origin": "4",
      "destiny": "4",
      "piece": "Red",
      "turn": "1"
    },
    {
      "origin": "5",
      "destiny": "5",
      "piece": "Yellow",
      "turn": "2"
    },
    {
      "origin": "4",
      "destiny": "4",
      "piece": "Red",
      "turn": "1"
    },
    {
      "origin": "5",
      "destiny": "5",
      "piece": "Yellow",
      "turn": "2"
    }
  ]
}

```

Figura 4.2: JSON de Movimentos de um Jogo

4.2.3 Conexão Entre Dois Jogadores

A gestão de toda a lógica de conexão entre jogadores é feita utilizando os serviços do Unity como Authentication, Lobby, Relay e Netcode for Game-Objects, conceitos aprendidos através de [Monkey, 2022].

1. Autenticação do jogador

Inicialmente, o Unity Services é inicializado e o jogador é autenticado de forma anónima. Esta autenticação é necessária para permitir o uso dos restantes serviços de rede do Unity.

2. Criação de um Lobby (*Host*)

Quando um jogador decide hospedar a partida, é criado um servidor Relay que gera um código de conexão. Esse código é adicionado como dado público no lobby. Em seguida, o jogador inicia o modo servidor, assumindo o papel de *host* da sessão. Tendo criado um código para o lobby, é apresentada ao utilizador a interface da figura 4.3.

Além disso, um processo de "heartbeat" é executado a cada 15 segundos para manter o lobby ativo, enviando um sinal de atividade para os serviços.

3. Entrada no Lobby (Cliente)

Para entrar num lobby existente, o jogador insere o código correspondente na caixa de texto. A aplicação recupera o código do Relay associado e conecta o jogador ao servidor, iniciando a sessão como cliente.

4. Sincronização

O Netcode for GameObjects foi usado para garantir a consistência e atualização em tempo real do estado do jogo, notificando os clientes quando um objeto é mudado. Já o Relay atua como servidor intermediário, permitindo a comunicação entre dispositivos mesmo em redes privadas ou atrás de Network Address Translation (NAT)/firewalls, sem necessidade de endereços IP públicos.

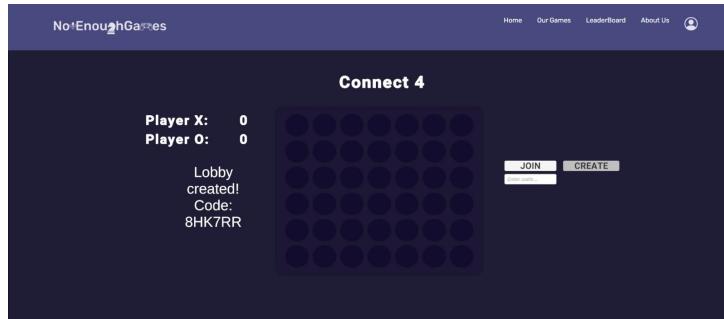


Figura 4.3: Quatro em Linha em Modo de Dois Jogadores

4.2.4 Inteligência Artificial

O algoritmo Monte Carlo Tree Search (MCTS) é constituído por quatro fases principais:

Seleção - A partir do nó raiz, que representa o estado inicial do jogo ou problema, o algoritmo percorre a árvore de decisões escolhendo os nós, que são pontos de decisão representando diferentes estados possíveis. Cada nó corresponde a um estado específico e está conectado aos seus nós filhos, que representam os estados alcançados após aplicar possíveis ações. Durante a seleção, o Monte Carlo Tree Search (MCTS) escolhe os nós com melhor valor Upper Confidence Bound for Trees (UCT), uma fórmula que equilibra a exploração de novos caminhos com os resultados obtidos anteriormente, até chegar a um nó folha, definida como

$$\text{UCT} = \frac{Q_i}{N_i} + c \cdot \sqrt{\frac{\ln(N_p)}{N_i}},$$

Onde:

- Q_i é a soma dos resultados do nó i ,
- N_i é o número de visitas ao nó i ,
- N_p é o número de visitas ao nó pai,
- c é a constante de exploração (utilizado $\sqrt{2}$).

O valor de UCT é calculado de forma a que nós com valores mais elevados sejam considerados melhores, pois indicam um bom equilíbrio entre explorar

ações pouco testadas e aquelas que já mostraram bons resultados. Tipicamente, o valor de UCT pode variar bastante, mas geralmente é positivo e sua magnitude depende dos parâmetros do problema e do número de visitas dos nós. O termo de exploração cresce com o inverso do número de visitas ao nó, incentivando a visita a nós menos explorados.

Expansão - Se o jogo ainda não terminou naquele estado, o nó é expandido com todos os movimentos legais possíveis a partir desse ponto.

Simulação - A partir do novo estado, simula-se uma partida aleatória até ao fim. O resultado da simulação (vitória, derrota ou empate) é usado como *feedback*.

Retropropagação - Os resultados são propagados de volta pelos nós visitados, atualizando o número de visitas e a soma dos resultados de cada nó.

Este ciclo é executado repetidamente dentro de um limite de tempo previamente definido. O número de simulações realizadas está diretamente condicionado por esse tempo, permitindo a criação de níveis de dificuldade distintos: quanto menor o tempo, menos simulações são feitas e menor tende a ser a qualidade da decisão tomada pelo algoritmo. Os tempos escolhidos em cada dificuldade para os jogos são apresentados na tabela 4.1.

Tabela 4.1: Tempos Típicos para os Níveis de Dificuldade em Cada Jogo

Jogo	Fácil (s)	Médio (s)	Difícil (s)
Jogo do Galo	0.00075	0.002	0.1
Quatro em Linha	0.004	0.05	0.75

O algoritmo MCTS foi estudado a partir de [AI e Games, 2018]. Ao desenvolver mais conhecimentos sobre este tema, foi adaptado o código de [Coder, 2022] para os jogos da plataforma e para a linguagem a ser utilizada (C#).

Os jogos funcionam com dois jogadores que jogam alternadamente. Em cada turno, a inteligência artificial analisa o estado atual do jogo e utiliza o algoritmo para simular as possíveis jogadas, escolhendo a melhor opção com base no tempo de procura que teve disponível. Após a decisão da IA, o movimento é aplicado ao tabuleiro, e a vez passa para o adversário. Esse processo repete-se até que o jogo termine.

4.2.5 Fluxo do Jogo

O jogo é preparado com base nas *flags* que indicam o contexto contra IA, contra outro jogador ou em modo de revisão.

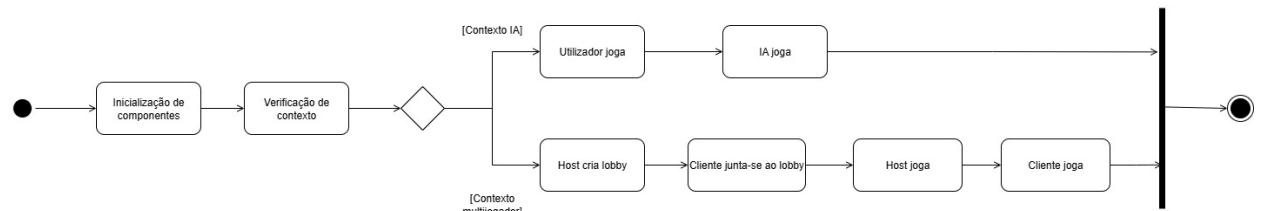


Figura 4.4: Diagrama de atividades da partida

Após a validação de cada jogada, verifica-se se esta conduz ao fim do jogo. Caso não leve a um estado terminal, o fluxo repete-se de forma cíclica.

4.3 Implementação do *Backend*

O *backend* foi implementado com recurso à linguagem PHP para o servidor e MySQL como sistema de gestão de base de dados.

4.3.1 Estrutura Geral

O *backend* está organizado em múltiplos ficheiros PHP, tais como autenticação, página inicial, jogo e perfil. Cada um corresponde a um *endpoint* que poderá ser acedido via URL e configurado para aceitar pedidos HTTP do tipo POST ou GET, dependendo da ação a executar.

4.3.2 Processamento de Dados

Os dados recebidos pelo *backend* são lidos a partir do corpo do pedido HTTP e interpretados em formato JSON. Para garantir integridade e evitar problemas, como injeção SQL, todas as interações com a base de dados utilizam *prepared statements*.

Além disso, dados sensíveis como palavras-passe são armazenados com um *hash* seguro. Desta forma, mesmo que a base de dados seja comprometida, a confidencialidade dos dados dos utilizadores é preservada.

4.3.3 Integração com Unity e HTML

No Unity, a comunicação com o *backend* é feita através de pedidos HTTP usando UnityWebRequest. As requisições POST, como o envio de resultados, movimentos ou início de partidas, são realizadas através da serialização dos dados em JSON antes de os enviar para o servidor. Já as requisições GET são feitas diretamente a partir da URL, que trata a resposta e altera dinamicamente a cena com base nas informações recebidas.

Quando a página é carregada, a primeira cena a ser exibida é o menu principal, responsável por comunicar com o servidor, decidir qual jogo será instanciado e em que contexto (IA, multijogador ou revisão). Para guardar informações entre cenas, como a dificuldade da IA, foi utilizado o sistema PlayerPrefs, que permite armazenar dados localmente.

As figuras 4.5 e 4.6 apresentam os jogos contidos na página HTML.



Figura 4.5: Interface do Jogo do Galo no Website

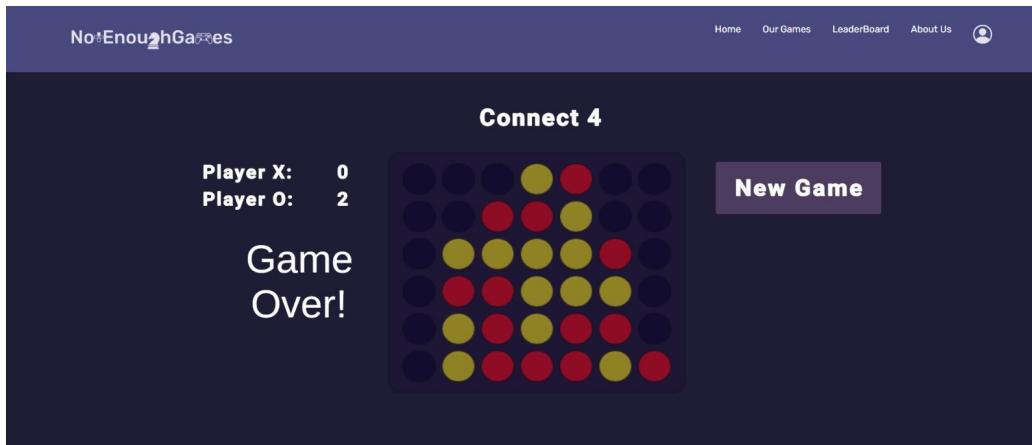


Figura 4.6: Interface do Quatro em Linha no Website

4.3.4 Estrutura da Base de Dados

A estrutura inclui tabelas para utilizadores, jogos, partidas e jogadas, interligadas por chaves primárias e estrangeiras. Foram definidos índices para acelerar operações frequentes, como a consulta de resultados. Segue-se um exemplo da tabela das partidas na figura 4.7.

	matchId	maxUsers	dhInicio	dhFim	title	winner
	65	2	2025-07-03 22:54:36	2025-07-03 22:54:52	TicTacToe	draw
	66	2	2025-07-03 22:58:42	2025-07-03 22:58:51	TicTacToe	draw
	67	2	2025-07-03 22:59:35	2025-07-03 22:59:57	Connect4	1
	68	2	2025-07-03 23:01:23	2025-07-03 23:01:36	TicTacToe	1
	69	2	2025-07-03 23:05:22	2025-07-03 23:05:50	TicTacToe	2

Figura 4.7: Base de Dados das Partidas

A base de dados também regista a pontuação de um utilizador para determinado jogo. Esta é utilizada para ser exibida no perfil ou realizar o cálculo dos quadros de honra, realizado a partir da fórmula presente em [Tornfy, 2023]. Ambas as funcionalidades são atingidas através de *prepared statements* realizados pelo PHP.

Capítulo 5

Validação e Testes

Este capítulo apresenta a validação prática da aplicação desenvolvida, demonstrando como os requisitos definidos foram cumpridos e analisando o desempenho da solução em diferentes cenários. A validação incluiu testes funcionais, testes com utilizadores reais e uma análise detalhada do comportamento da Inteligência Artificial nos dois jogos distintos.

Na secção 5.1 são descritas as funcionalidades essenciais implementadas. Esta secção também aborda a estabilidade do sistema, a segurança na comunicação com o servidor e o uso da interface.

As secções 5.2 e 5.3 descrevem os testes realizados nos jogos implementados, Jogo do Galo e Quatro em Linha, com foco na avaliação da IA em diferentes níveis de dificuldade. Foram analisados os resultados das partidas, permitindo verificar a adequação do algoritmo Monte Carlo Tree Search (MCTS) e a sua capacidade de adaptação a cada jogo.

Por fim, a secção 5.4 apresenta os resultados do inquérito realizado junto a um grupo de utilizadores com diferentes perfis. O *feedback* obtido contribuiu para os últimos ajustes da aplicação, nomeadamente ao nível da experiência e das mensagens apresentadas durante o jogo.

5.1 Funcionalidades Essenciais

Verificou-se que todos os requisitos essenciais definidos no início do projeto foram concretizados com sucesso. A aplicação permite a autenticação de utilizadores, o jogo contra a Inteligência Artificial com diferentes níveis de dificuldade (figura 5.1), partidas entre diferentes jogadores online, atribuição

automática de conquistas consoante o desempenho (figura 5.2), reconstrução e visualização de partidas anteriores e um quadro de honra ordenado por pontuações (figura 5.3).

As conquistas funcionam como forma de recompensar o progresso dos jogadores e incentivar a continuidade da utilização da plataforma. Estas são atribuídas automaticamente com base em critérios definidos no desempenho dos utilizadores. Algumas distinções incluem, por exemplo, a “*First Match*”, atribuída após a primeira partida realizada, e a “*Breakthrough Win*”, desbloqueada com a primeira vitória. Existem também conquistas ligadas à frequência de jogo, como a “*Grinder*” (10 partidas num só dia) ou a “*Veteran*” (100 partidas realizadas).

Ao nível competitivo, o sistema reconhece desempenhos de excelência com distinções ligadas ao posicionamento nos rankings: “*High Performer*” para o top 30, “*Elite Ranker*” para o top 10, “*Podium Legend*” para o top 3 e “*Invincible*” para quem atinge o primeiro lugar. Estas conquistas contribuem para reforçar a componente de competição da aplicação.

Todas as funcionalidades comunicam de forma segura com o servidor PHP, garantindo a integridade dos dados, e a interface cumpre os requisitos de simplicidade e clareza, proporcionando uma boa experiência ao utilizador.

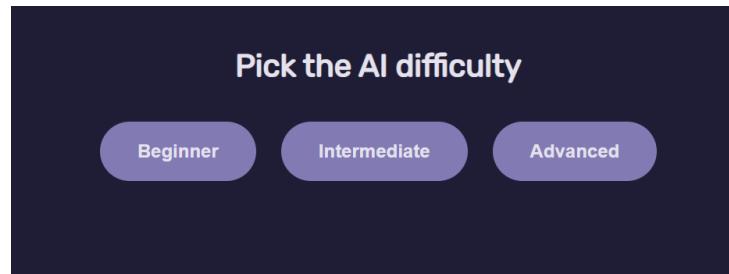


Figura 5.1: Escolha da Dificuldade da IA



Figura 5.2: Atribuição de Conquistas

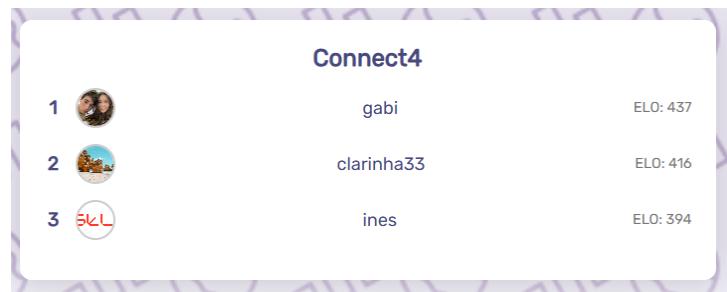


Figura 5.3: Quadro de Honra do Jogo do Galo

5.2 Jogo do Galo

O jogo do Galo foi utilizado como prova de conceito inicial para a integração de interação com o jogo, inteligência artificial e funcionalidades *online*, por ser um jogo mais simples. A IA mostrou um comportamento ideal, sendo praticamente imbatível no nível mais elevado e adotando estratégias menos rigorosas nos níveis de dificuldade mais baixos. Ao realizar dez jogos para cada dificuldade foram obtidos os resultados presentes na tabela 5.1.

Tabela 5.1: Resultados dos Testes do Jogo do Galo

Nível	Humano Ganha	IA Ganha	Empate
Beginner	8	1	1
Intermediate	6	2	2
Advanced	0	0	10

5.3 Quatro em Linha

O jogo Quatro em Linha trouxe desafios acrescidos ao desempenho da IA, dada a sua maior complexidade de espaços de estados. O algoritmo Monte Carlo Tree Search (MCTS) revelou-se eficaz na tomada de decisões, adaptando-se ao tempo de execução disponível para oferecer diferentes graus de dificuldade. Durante os testes, verificou-se que este jogo serviu como bom indicador da eficiência do algoritmo MCTS, com os utilizadores mais experientes a enfrentarem os níveis mais difíceis, obtendo resultados razoáveis. Para o jogo Quatro em Linha também se realizaram testes dos quais os resultados encontram-se na seguinte tabela 5.2, sendo que foram feitos dez testes de acordo com cada dificuldade.

Tabela 5.2: Resultados dos Testes do Quatro em Linha

Nível	Humano Ganha	IA Ganha	Empate
Beginner	10	0	0
Intermediate	6	4	0
Advanced	0	10	0

5.4 Inquérito aos Jogadores

Com o objetivo de recolher *feedback* real sobre a experiência de utilização, foi distribuído um formulário de inquérito após os testes com utilizadores. Este questionário abordou temas como o comportamento da inteligência artificial, a clareza das funcionalidades de perfil e conquistas, o interesse nos jogos oferecidos e a satisfação geral com a plataforma. Algumas das perguntas

contidas no formulário e respetivas percentagens de respostas apresentam-se nas figuras 5.4, 5.5, 5.6 e 5.7.

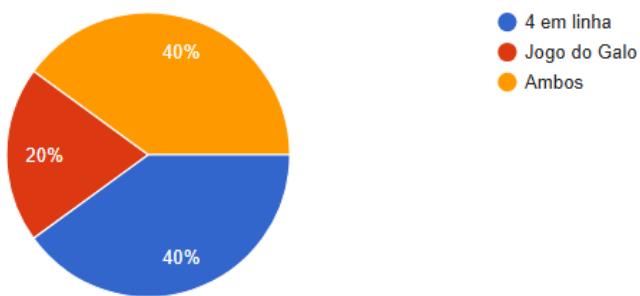


Figura 5.4: Preferência dos utilizadores pelos jogos disponíveis na plataforma

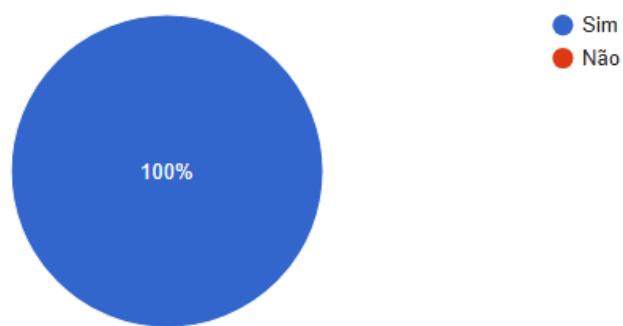


Figura 5.5: Percentagem de utilizadores que acharam a experiência de jogo clara e intuitiva

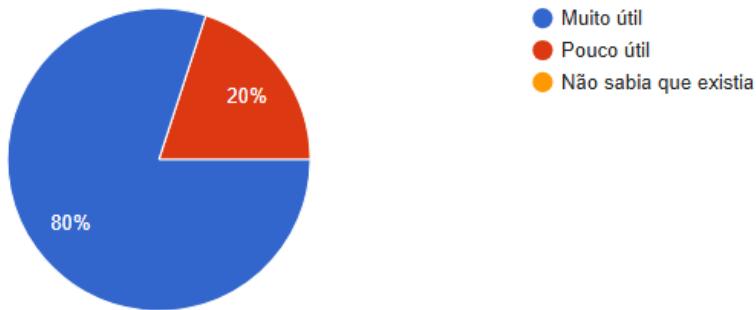


Figura 5.6: Avaliação da funcionalidade de revisão de jogos.

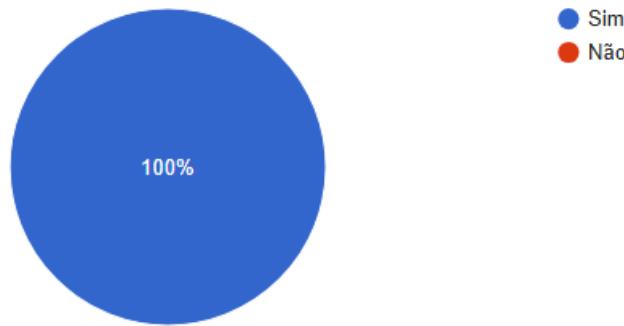


Figura 5.7: Correspondência entre dificuldade escolhida e expectativas dos jogadores

Participaram dez utilizadores no formulário dos quais 40% eram homens e 60% mulheres. Do ponto de vista etário e de experiência, 60% eram jovens com experiência prévia em jogos digitais, enquanto os restantes 40% correspondiam a adultos com menos contacto com este tipo de plataformas. As respostas evidenciaram elevada satisfação com a experiência geral da plataforma, em especial no que diz respeito à simplicidade da interface e à expectativa da dificuldade da IA. Algumas sugestões foram incorporadas, como melhorias na interface e mensagens mais claras de erro e sucesso durante o jogo.

Capítulo 6

Conclusões e Trabalho Futuro

A necessidade de proporcionar uma plataforma para jogos clássicos e a análise das suas jogadas esteve na origem deste projeto. Pretendeu-se criar um ambiente onde os utilizadores pudessem competir entre si ou contra algoritmos de inteligência artificial, ao mesmo tempo que acompanhavam o seu progresso e desempenho.

Para concluir os objetivos, foi desenvolvida uma aplicação que integra uma componente gráfica interativa, uma base de dados para gestão da informação, e jogos com inteligência artificial com níveis adequados de dificuldades. A estrutura apresenta suporte a múltiplos jogos, modos de jogo distintos e funcionalidades de perfil, conquistas e quadros de honra, utilizando tecnologias como Unity, HTML, CSS, JavaScript, PHP e MySQL, bem como ferramentas como XAMPP e serviços de rede do Unity.

A plataforma revelou-se funcional e estável, tendo cumprido todos os requisitos definidos. As respostas recolhidas através do formulário indicaram uma experiência de utilização positiva, com destaque para a experiência com os jogos e o comportamento desafiante da IA. A utilização do algoritmo MCTS permitiu alcançar um equilíbrio entre desempenho e realismo no comportamento da IA.

Como trabalho futuro, prevê-se a introdução de novos jogos, um sistema de edição de perfil, socialização com outros utilizadores e tratamento de erros como, por exemplo, tempo limite de jogo e jogada. Poderá ainda ser explorada a implementação de uma aplicação móvel.

Apêndice A

Regras dos Jogos

A.1 Jogo do Galo

Objetivo

O objetivo do Jogo do Galo é ser o primeiro jogador a formar uma linha com três símbolos iguais numa das direções possíveis: horizontal, vertical ou diagonal.

Componentes

- Um tabuleiro de três linhas por três colunas;
- Dois jogadores;
- Cada jogador escolhe um símbolo, X ou O.

Regras

1. Os jogadores jogam alternadamente. Em cada turno, o jogador deve marcar uma das casas vazias com o seu símbolo. Não é permitido jogar numa casa que já tenha sido marcada anteriormente;
2. Vence o jogador que conseguir formar uma linha com três dos seus símbolos. Essa linha pode estar disposta na horizontal, na vertical ou na diagonal;
3. Se todas as casas forem preenchidas e nenhum dos jogadores tiver conseguido formar uma linha de três símbolos, a partida termina com um empate.

A.2 Quatro em Linha

Objetivo

O objetivo do 4 em Linha é ser o primeiro jogador a alinhar quatro peças da sua cor de forma consecutiva na horizontal, vertical ou diagonal.

Componentes

- Um tabuleiro vertical com 6 linhas por 7 colunas;
- Dois jogadores;
- Cada jogador tem um conjunto de peças de uma cor distinta.

Regras

1. O tabuleiro é colocado na vertical, de forma que as peças inseridas nas colunas "caiam" até à posição mais baixa disponível. Isso simula a ação da gravidade: uma peça só pode ocupar a posição mais baixa da coluna em que for jogada ou pousar em cima de outra peça já existente;
2. Os jogadores jogam alternadamente. Em cada turno, um jogador escolhe uma das colunas e deixa cair uma das suas peças nessa coluna. Não é possível jogar numa coluna que já esteja completamente cheia;
3. O jogo continua com os jogadores alternando jogadas até que um dos dois consiga alinhar quatro peças consecutivas da sua cor. Esse alinhamento pode ser horizontal, vertical ou diagonal;
4. Se todas as colunas estiverem cheias e nenhum jogador tiver conseguido alinhar quatro peças, a partida termina com um empate.

Bibliografia

[Aficionado, 2025] Aficionado (2025). Chess.com. <https://www.chess.com/>. Consultado a 10-07-2025.

[AI e Games, 2018] AI e Games (2018). Ai 101: Monte carlo tree search. <https://www.youtube.com/watch?v=1hFXKNyAOQA>. Consultado a 30-05-2025.

[Bokarev, 2025] Bokarev, T. (2025). Tabletopia. <https://tabletopia.com/>. Consultado a 10-07-2025.

[Carriço e Rodrigues, 2025] Carriço, A. e Rodrigues, G. (2025). Notenoughgames. <https://notenoughgames-e8dke0edddhkckfc.spaincentral-01.azurewebsites.net/>. Consultado a 14-07-2025.

[Coder, 2022] Coder, R. (2022). Beating connect 4 with monte carlo tree search! | explanation + code. <https://www.youtube.com/watch?v=EB-NJtNERBQ>. Consultado a 15-06-2025.

[Isabelli e Colin, 2025] Isabelli, G. e Colin, E. (2025). Board game arena. <https://en.boardgamearena.com/>. Consultado a 10-07-2025.

[Monkey, 2022] Monkey, C. (2022). Making a multiplayer game? join your players with lobby! <https://www.youtube.com/watch?v=-KD1EBfCBiU&t=1169s>. Consultado a 16-04-2025.

[Neto e Silva, 2004] Neto, J. P. e Silva, J. N. (2004). *Jogos Matemáticos* *Jogos Abstratos*.

[Oamen, 2023] Oamen, D. (2023). How to create a tic tac toe game in unity - full tutorial. https://www.youtube.com/watch?v=3sc0HXEKn_0. Consultado a 13-04-2025.

- [Ravikiran, 2025] Ravikiran, A. S. (2025). How to run a php file using xampp: A step by step guide. <https://www.simplilearn.com/tutorials/php-tutorial/php-using-xampp>. Consultado a 19-04-2025.
- [Russell e Norvig, 2010] Russell, S. J. e Norvig, P. (2010). *Artificial Intelligence A Modern Approach*. Wiley Series in Agent Technology. Pearson, 3 edition.
- [Technologies, 2017] Technologies, U. (2017). Building and running a webgl project. <https://docs.unity3d.com/560/Documentation/Manual/webgl-building.html>. Consultado a 28-03-2025.
- [Tornfy, 2023] Tornfy (2023). Rating elo. <https://tornfy.com/RatingEloTornfy>. Consultado a 05-06-2025.